Feature Selecting Hierarchical Neural Network for Industrial System Health Monitoring: Catching Informative Features with LASSO

Gabriel Michau[1], Manuel Arias Chao[2], and Olga Fink[3]

[1,2,3] *Zurich University of Applied Sciences, Rosenstr. 3, Winterthur, 8401, Switzerland*
gabriel.michau@zhaw.ch
manuel.ariaschao@zhaw.ch
olga.fink@zhaw.ch

ABSTRACT

Industrial System Health Monitoring relies usually on the monitoring of well-designed features. This requires both, the engineering of reliable features and a good methodology for their analysis. If traditionally, features were engineered based on the physics of the system, recent advances in machine learning demonstrated that features could be automatically learned and monitored. In particular, using Hierarchical Extreme Learning Machines (HELM), based on random features, very good results have already been achieved for health monitoring with training on healthy data only.

Yet, although very useful and mathematically sound, random features have little popularity as they contradict the intuition and seem to rely on luck. This tends to increase the "black-box" effect often associated with Machine Learning. To mitigate this, in this paper, we propose to modify the traditional HELM architecture such that, while still relying on random features, only the most useful features among a large population will be selected.

Traditional HELM are made of stacked contractive auto-encoders with ℓ_1- or ℓ_2-regularisation and of a classifier as last layer. To achieve our objective, we propose to opt for expanding auto-encoders instead, but trained with a strong Group-LASSO regularization. This Group-LASSO regularisation fosters the selection of as few features as possible, making the auto-encoder in reality (or in testing condition) contractive. This deterministic selection provides useful features for health monitoring, without the need of learning or manually engineering them.

The proposed approach demonstrates a better performance for fault detection and isolation on case studies developed for HELM evaluation.

1. INTRODUCTION

Context and Related Works: Recently, the number of industrial assets equipped with condition monitoring devices has been rapidly increasing. Decreased costs for data transmission and storage are increasingly enabling real-time access to high resolution signals that have previously partly only been used within system control of critical systems. Often, refurbishments of complex systems are used as an opportunity to install or upgrade condition monitoring systems and make the information on system condition available in real time for interventions and system health management. One of the further developments triggered by the industrial internet of things is that the information is not only collected centrally but that the connected systems can exchange the information of the operating conditions and system states in a decentralized way, enabling thereby transfer of operating experience and fault types.

Large amounts of condition monitoring sensors enable also measuring secondary effects of faults, in cases where primary effects are either not accessible or not measurable. This enables detecting and predicting faults that have not been observable and detectable before.

Even though physics of failure approaches provide detailed and well understood fault pattern evolution and prediction (Pecht & Jie Gu, 2009; Tinga & Loendersloot, 2014), the models are often only available for a limited number of systems and components. While data-driven prognostics and health management (PHM) approaches have been exceedingly growing in last years, particularly with the use of deep learning approaches (Gugulothu et al., 2017; Ince, Kiranyaz, Eren, Askar & Gabbouj, 2016; Michau, Palmé & Fink, 2017; Sateesh Babu, Zhao & Li, 2016), new challenges are

arising for PHM applications based on high-dimensional, high-frequency condition monitoring data.

In this context, feature engineering which requires expert knowledge (Vachtsevanos, Lewis, Roemer, Hess & Wu, 2006) can be very time consuming, especially in high dimensional condition monitoring applications. Yet, it involves systems engineers in the PHM design process and enables a better interpretability of the results. Therefore, it is still often favoured for real applications. Contrary to the feature engineering approach, feature learning approaches do not rely on expert knowledge, they enable end-to-end learning and are much faster to implement and to transfer to new applications (Michau, Yang, Palmé & Fink, 2017).

Baseline: Recently, a deep learning approach for a combined fault detection and isolation has been proposed (Michau, Palmé & Fink, 2017), based on stacked autoencoders a one-class classifier representing the system health, namely Hierarchical Extreme Learning Machines (HELM). The proposed approach is trained on healthy data only and demonstrates a robust performance in detecting abnormal conditions and isolating the sources of such abnormalities. It does neither require any prior knowledge on faults, nor labels for classes. As such, it does not aim at classifying different unhealthy conditions nor separate inputs, neither in the feature space, nor in the output layer. The primary goal is to learn a representation of "healthy" system conditions and to compare the newly observed conditions to those known as being "healthy". The matter of making the method robust to more operating conditions than those experienced by a single unit and enlarging the representation of "healthy" system conditions, has been discussed from a fleet perspective in (Michau, Palmé & Fink, 2018).

As the features in HELM are random (but the weights to these features are not), it is difficult to link them to any physical meaning and interpretation. Even though HELM is achieving convincing results on fault detection, this randomness lowers acceptance: Only within the fault isolation module could the patterns for different fault types be analysed by the experts.

Contributions: In the present paper, we propose a new hierarchical neural network that combines a fast and efficient end-to-end learning process with a feature selection process. The proposed approach extends the work discussed above (Michau, Palmé & Fink, 2017; Michau, Yang et al., 2017). By implementing feature selection among a large population of randomly drawn features, one might hope to select the most informative features. In this paper, we demonstrate that this approach provides informative features for health monitoring as the performance in fault isolation is increased compared to pure HELM applications. This deterministic selection of the best performing features, or selection of the "fittest", reduces the randomness, improves the performance and aims at contributing to increasing the acceptance of data-driven algorithms in PHM applications.

Similarly to HELM, the approach has a two-level hierarchical structure. At the first level, a population of random features is generated, based on the framework of Extreme Learning Machines. Yet, only the best subset of features with respect to the auto-encoding of the healthy input is selected. At the second level, the learned features are used in a one-class classifier extreme learning machine in order to estimate if the data is healthy (conform to training) or is experiencing deviations from the healthy conditions observed so far.

The rest of the paper is organised as follows. First, in Section 2, the historic motivations for hierarchical structures are presented. Then, in Section 3, the new methodology, Feature Selecting Hierarchical Neural Network (FSHNN) is detailed while put in perspective with other similar approaches. Last, in Section 4, FSHNN is applied to fault detection and isolation, the results are compared to HELM and discussed.

Notation: In the following, we denote by $\|\cdot\|$ the element-wise norm for matrices: e.g.,

$$\begin{aligned} \|X\|_p &= \sqrt[p]{\textstyle\sum_{ij} |X_{ij}|^p} \\ \|X\|_{p,q} &= \sqrt[q]{\left(\textstyle\sum_j \sqrt[p]{\textstyle\sum_i X_{ij}^p}\right)^q} \end{aligned} \tag{1}$$

2. HIERARCHICAL DEEP NEURAL NETWORKS

In the development of neural networks, it is well-known that deeper structures often lead to better results. In fact, the number of parameters to learn grows exponentially with the number of layers, giving the networks more and more degrees of freedom to adapt to complex models. This "deepening" is at the expense of computational efficiency, recently largely mitigated by the rise of computational power. Yet, one of the limitations with achieving deeper structures is the difficulty to train the network: The loss of the network, that is, its distance to the objective, is computed at the last layer and usually fed backward in the network with back propagation. Back propagation relies on the chain rule for derivatives, to compute at each layer the gradient[1], with respect to the weights at that layer, based on the gradient computed at the next layer. As a learning process, this happens to be quite often inefficient, as the impact of the "earliest" layers on the loss are mitigated by all the layers coming next. This makes the gradient often vanish, sometimes explodes and necessitates high numbers of iterations to reach convergence of the learning process. In addition, in multi-objective optimisation, it appears that convergence is usually ensured for learning steps smaller than a value based on the Lipschitz constant of the objective function (Briceño-Arias, Combettes, Pesquet & Pustelnik,

[1] or its generalisation in case of non-differentiable functions as sub-gradient or proximity operator

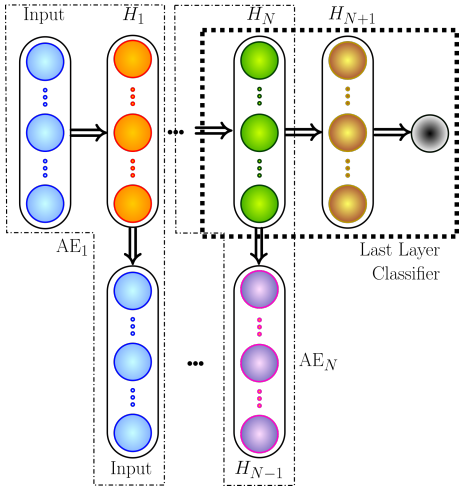

Figure 1. **Hierarchical Neural Networks**. The architecture consists in stacked single layered neural network where each hidden layer is the input of the next layer. It mimics traditional deep architecture while allowing for sequencing and independent training, one layer at a time.

2011). For deep structures, computing this constant would be cumbersome, requiring the computation of the gradient, and traditionally, an arbitrary very small learning step is chosen instead. This sub-optimality in the choice of the learning step increases even more the number of iterations needed to achieve convergence.

A solution to such limitations is the sub-division of the network in smaller structures, each trained with their own objectives. For, example, one can use a set of single layered feed forward neural networks (SLFN) and order them hierarchically such that each network uses as inputs the hidden layer of the previous one. These are the Hierarchical Neural Networks (HNN). Conceptually, the chain composed of the input, the successive hidden layers and the output mimics perfectly the traditional deep architecture for feed-forward neural networks.

In theory, the first networks in hierarchical structure can be designed and trained with respect to any objective. Yet, the idea in using HNN is to avoid the computation of the final objective gradient at earliest layers of the network. Therefore, it is natural to look for architectures where the first neural networks would be trained in an unsupervised way while the last one would be trained for the objective. In many works, HNN are composed of a succession of auto-encoders and of a last regression or classification layer. The concept of HNN with auto-encoders is illustrated in Figure 1.

It has been demonstrated that, given enough neurons, it is possible to approximate any function with a SLFN where the weights between the input and the hidden layer are drawn randomly and only the weights between the hidden layer and the output are learned (G.-B. Huang, Chen, Siew et al., 2006).

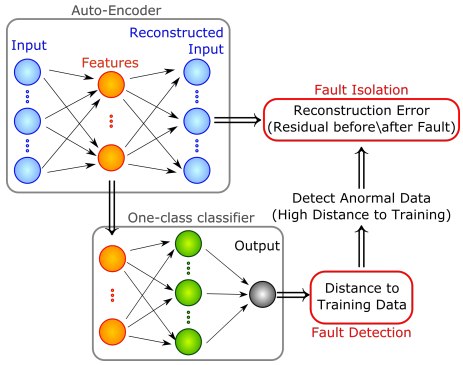

Figure 2. **HELM for Fault Detection and Isolation**. It consists of a first layer for feature learning and of a second layer, using the features for health monitoring. Features are used for identification once abnormalities have been detected.

This is a computationally much simpler hence faster optimisation problem to solve, which explain the strong interest such networks raised. This kind of SLFN are called Extreme Learning Machines (ELM). Combining both ideas of HNN and ELM leads to the architecture commonly known as Hierarchical Extreme Learning Machines (HELM) (Tang, Deng & Huang, 2016).

Hierarchical Extreme Learning Machines have been applied in many fields (G. Huang, Huang, Song & You, 2015; G.-B. Huang, Wu & Wunsch, 2018; Man & Huang, 2016), including PHM (Michau, Palmé & Fink, 2017, 2018; Michau, Yang et al., 2017; Yang, Fink & Palmé, 2016). In PHM, the interest for Hierarchical Networks lies in that they actually imitate the traditional approaches: The auto-encoders, by learning the important components and relationships between the input signals are doing feature learning while the last layer, by combining the feature to assess the conformity of the data with the training is used as an health monitoring tool. In particular, in (Michau, Palmé & Fink, 2017), it has been shown that with the right architecture, HELM can be used in an unsupervised framework. Instead of the traditional fault classification, a one class classifier is used as a last layer and the HELM is trained with healthy data only. This methodology demonstrated excellent ability to detect abnormal operating conditions and to isolate the source of abnormality thanks to the auto-encoders. This framework is illustrated in Figure 2.

3. FEATURE SELECTING HIERARCHICAL NEURAL NETWORK

3.1. From Single Layer Networks to Hierarchical Neural Networks

Traditional Single Layer Feed Forward Network consists in performing the following operation:

$$\boldsymbol{Y} = \boldsymbol{B} \cdot g\left(\boldsymbol{A} \cdot \boldsymbol{X}\right), \tag{2}$$

where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are the input and output, of size D_i and D_o dimensions, K samples. $\boldsymbol{A}$ and $\boldsymbol{B}$ are respectively the input and output weights of size $H \times D_i$ and $D_o \times H$, H being the number of neurons in the network. g is the activation function. In addition to weights, biases can be considered and simply consists in concatenating to $\boldsymbol{X}$ and then to $g(\boldsymbol{A} \cdot \boldsymbol{X})$ an additional dimension filled with ones. The biases are then the corresponding elements in $\boldsymbol{A}$ and $\boldsymbol{B}$.

In the case of Extreme Learning Machines, $\boldsymbol{A}$ is drawn randomly. Given an training set $\boldsymbol{X}^{\text{Train}}$ and a target output T, training an ELM consists therefore in solving the following optimization problem:

$$\hat{\boldsymbol{B}} = \underset{\boldsymbol{B}}{\text{Argmin}} \|\boldsymbol{B}\boldsymbol{H} - T\|_u^{\sigma_1} + \lambda \|\boldsymbol{B}\|_v^{\sigma_2} \tag{3}$$

where $\boldsymbol{H} = g\left(\boldsymbol{A} \cdot \boldsymbol{X}^{\text{Train}}\right)$. In the vast majority of works, $u = \sigma_1 = 2$: the ℓ_2 norm is continuous and easily differentiable while also measuring the euclidean distance between target and output. The second term of the objective function is called the regularisation term. It models properties of the learned variable, that one wish to encourage. A common choice is that of also using $v = \sigma_2 = 2$. This aims at avoiding diverging coefficient in $\boldsymbol{B}$. In case of correlated variables, this avoids in particular that their coefficient compensates each others, thus, over-fit the problem. This corresponds therefore to the following Ridge Regression problem:

$$\hat{\boldsymbol{B}} = \underset{\boldsymbol{B}}{\text{Argmin}} \|\boldsymbol{H}\boldsymbol{B} - T\|_2^2 + \lambda \|\boldsymbol{B}\|_2^2 \tag{4}$$

which has a closed form solution:

$$\hat{\boldsymbol{B}} = \left(\lambda \cdot \mathbb{I} + \boldsymbol{H}^\top \boldsymbol{H}\right)^{-1} \boldsymbol{H}^\top T. \tag{5}$$

3.2. Auto-encoders in FSHNN

In previous works (Michau, Palmé & Fink, 2017; Michau, Yang et al., 2017; Tang et al., 2016), it has been suggested to use the ℓ_1 norm for the regularisation of the auto-encoders, that is, $v = \sigma_2 = 1$. The idea is that, by encouraging sparsity, and therefore as few features-to-signals connections as possible, the obtained features would be of higher quality. In fact, a good feature is expected to be very informative on part of the signal. Yet, in HELM, the features are randomly drawn and this assumption is hard to verify. Therefore, in practice, it often appears that the regularisation parameter that would indeed bring strong sparsity need to be high, at the expense of the reconstruction error (measuring the auto-encoder quality). In consequence, best results of the final HELM are achieved for low sparsity in the auto-encoder, in contradiction with the premise on which the idea of the ℓ_1 norm has been developed. In addition, as the input weights are drawn randomly, the features represented by the hidden neurons $g(\boldsymbol{A} \cdot \boldsymbol{X})$ are *a priori* meaningless. They only make sense once the output weights $\boldsymbol{B}$ have been learned, in that their connection to the reconstructed input indicates the participation of each feature to the signal. This *a posteriori* interpretation of the random feature is conceptually unsettling.

With these two points in mind, we propose in this work to impose group-sparsity on the output weights $\boldsymbol{B}$. That is, a regularisation that will shut-down all connections from features that are the least participating to the signal reconstruction. Although the features are randomly generated, in the final solution, only the most meaningful features will be selected and used in the Feature-Selecting HNN (FSHNN). This problem is referred to in the litterature as the Best Subset Selection Problem (Friedman, Hastie & Tibshirani, 2001; Hocking & Leslie, 1967) and consists in solving

$$\hat{\boldsymbol{B}} = \underset{\boldsymbol{B}}{\text{Argmin}} \|\boldsymbol{H}\boldsymbol{B} - X\|_2^2 + \lambda \|\boldsymbol{B}\|_{2,0} \tag{6}$$

Problem (6) is not a linear nor a convex problem. Recent advances have shown that it is solved with Mixed Integer Optimisation (Bertsimas, King & Mazumder, 2016). Yet, in convex optimisation, the Group-LASSO is usually used as a surrogate for solving the problem. Studies have shown that it is actually very competitve with other methods (Hastie, Tibshirani & Tibshirani, 2017). It consists, in the case of our auto-encoder, in solving:

$$\hat{\boldsymbol{B}} = \underset{\boldsymbol{B}}{\text{Argmin}} \|\boldsymbol{H}\boldsymbol{B} - X\|_2^2 + \lambda \|\boldsymbol{B}\|_{2,1} \tag{7}$$

where

$$\|\boldsymbol{B}\|_{2,1} = \sum_j \sqrt{\sum_i B_{i,j}^2}. \tag{8}$$

The $\ell_{2,1}$-norm has a proximity operator (Pustelnik, Chaux & Pesquet, 2009):

$$\text{prox}_{\gamma \ell_{2,1}}(X) = \begin{cases} \left(1 - \frac{\gamma}{\sqrt{\sum |x|^2}}\right) x & \text{if} \sqrt{\sum |x|^2} \geq \gamma \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{9}$$

Therefore Problem (7) can be solved thanks to the FISTA algorithm described in Algorithm 1, chosen here for its efficiency and computational simplicity (Beck & Teboulle, 2009; Chambolle, Dossal et al., 2014).

Once Problem (7) is solved, we identify the unused features (empty columns in $\boldsymbol{B}_k$) and remove their corresponding elements from $\boldsymbol{A}$ and $\boldsymbol{B}$. Then, the optimal $\boldsymbol{B}$ for auto-encoding is obtained by solving one last time the ridge regression, using Equation (5).

3.3. The One-class classifier

As a last layer of the FSHNN, we use a one class classifier (Khan & Madden, 2009). As demonstrated in previous works (Michau, Palmé & Fink, 2017, 2018; Michau, Yang et al., 2017; Yang et al., 2016), once good features are identified,

Algorithm 1 FISTA

Input: $\boldsymbol{H}, \quad X, \quad \lambda \geq 0, \quad \delta \in]0,1[, \quad \epsilon \geq 0$

$\gamma \leftarrow \frac{\delta}{1+\|\boldsymbol{H}\|_2}$
$k \leftarrow 0, \quad \boldsymbol{B}_k \leftarrow 0, \quad \boldsymbol{y}_k \leftarrow 0,$
$t_k \leftarrow 1, \quad \text{Crit} \leftarrow \infty,$
while Crit $\geq \epsilon$ **do**
  $\boldsymbol{c}_k = \boldsymbol{y}_k - 2 \cdot \gamma \boldsymbol{H}^\top (\boldsymbol{H}\boldsymbol{y}_k - X)$
  $\boldsymbol{B}_{k+1} = \text{prox}_{\gamma\lambda\ell_{2,1}}(\boldsymbol{c}_k)$ ▷ (*cf.* Eq. (9))
  $t_{k+1} = \frac{1}{2}\left(1 + \sqrt{1+4t_k^2}\right)$
  $y_{k+1} = \boldsymbol{B}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{B}_k - \boldsymbol{B}_{k+1})$
  $\text{Crit} = \|\boldsymbol{B}_k - \boldsymbol{B}_{k+1}\|_2$
  $k = k + 1$

Output: $\boldsymbol{B}_k$

Algorithm 2 FSHNN Training

Input: $\lambda_{l_{21}}, \lambda_{AE}, \lambda_{1C} \geq 0, \quad N \in \mathbb{N}, \quad X^{\text{Train}}$

$\boldsymbol{X}_1 \leftarrow X^{\text{Train}}$
for $i = 1, \ldots, N$ **do** ▷ *Stacked FS AE ELM*
  Generate: $\boldsymbol{A}_i$ random weights
  $\boldsymbol{H}_i = g(\boldsymbol{A}_i \cdot \boldsymbol{X}_i)$
  $\boldsymbol{B}_i = \underset{\boldsymbol{B}}{\text{Argmin}} \|\boldsymbol{B} \cdot \boldsymbol{H}_i - \boldsymbol{X}_i\|_2^2 + \lambda_{l_{21}}\|\boldsymbol{B}\|_{2,1}$ ▷ (*cf.* Alg. 1)
  $c_{\boldsymbol{B}} \leftarrow$ index of non empty columns in $\boldsymbol{B}_i$
  $\boldsymbol{A}_i = \boldsymbol{A}_i[c_{\boldsymbol{B}}, :])$
  $\boldsymbol{H}_i = g(\boldsymbol{A}_i \cdot \boldsymbol{X}_i)$
  $\boldsymbol{B}_i = \underset{\boldsymbol{B}}{\text{Argmin}} \|\boldsymbol{B} \cdot \boldsymbol{H}_i - \boldsymbol{X}_i\|_2^2 + \lambda_{AE}\|\boldsymbol{B}\|_2^2$ ▷ (*cf.* Eq. (5))
  $\boldsymbol{X}_{i+1} = \boldsymbol{H}_i$ ▷ *Upper layer ELM*
$\boldsymbol{X}_{N+1} = [\boldsymbol{X}_{N+1}, \mathbb{I}_K]$
Generate: $\boldsymbol{A}_{N+1}$, random weights
$\boldsymbol{H}_{N+1} = g(\boldsymbol{A}_{N+1} \cdot \boldsymbol{X}_{N+1})$
$\boldsymbol{B}_{N+1} = \underset{\boldsymbol{B}}{\text{Argmin}} \|\boldsymbol{B} \cdot \boldsymbol{H} - 1\|_2^2 + \lambda_{1C}\|\boldsymbol{B}\|_2^2$ ▷ (*cf.* Eq. (5))

Output: $\{\boldsymbol{A}_i, \boldsymbol{B}_i\}_{1 \leq i \leq N+1}$

Algorithm 3 Running FSHNN

Input: $X, \quad \text{FSHNN}(\lambda_{AE},\ \lambda_{1C},\ N,\ X^{\text{Train}})$

$\boldsymbol{X}_1 \leftarrow X$
$\boldsymbol{X}_2 = g(\boldsymbol{A}_1 \cdot \boldsymbol{X}_1)$
$\boldsymbol{X}_{\text{rec}} = \boldsymbol{B}_1 \cdot \boldsymbol{X}_2$
for $i = 2, \ldots, N$ **do**
  $\boldsymbol{X}_{i+1} = g(\boldsymbol{A}_i \cdot \boldsymbol{X}_i)$
$\boldsymbol{X}_{N+1} = [\boldsymbol{X}_{N+1}, \mathbb{I}_K]$
$Y = \boldsymbol{B}_{N+1} \cdot g(\boldsymbol{A}_{N+1} \cdot \boldsymbol{X}_{N+1})$

Output: $Y, \boldsymbol{X}_{\text{rec}}$

the one-class classifier ELM is efficient at detecting any conditions that have not been represented in the training dataset. As such, it provides excellent detection results.

In our work, the one-class classification problem is formulated as a ridge regression problem (*cf.* Problem (4)), where the target is the value 1, representing healthy conditions. The distance between the output of the classifier in testing condition and the value 1 is monitored. If it increases compared to that measured during validation, the data is labelled as abnormal. More precisely, we set a fault-detection threshold as follows:

$$\text{Thrd} = \gamma \cdot \text{percentile}_p(|1 - Y^{\text{Val}}|) \tag{10}$$

where percentile_p is the pth-percentile function, p and $\gamma \geq 0$ are hyper-parameters, and Y^{Val} is the validation output. The class Z is then expressed as

$$Z_i^{\text{Test}} = \text{sgn}\left(\text{Thrd} - |1 - Y_i^{\text{Test}}|\right) \tag{11}$$

3.4. Training and Running FSHNN

The algorithms to train and run the FSHNN are described in Algorithm (2) and (3). N is the number of auto-encoders ($N = 1$ in our work). In the training, λ_H, the regularisation parameter for the Group-LASSO can either be set as an hyper-parameter, or, with few iterations, the value giving an *a priori* fixed number of features can be heuristically looked for. We implemented this second alternative.

4. FSHNN FOR FAULT DETECTION AND ISOLATION

In previous works (Michau, Palmé & Fink, 2017; Michau, Yang et al., 2017), extensive comparisons have demonstrated the very good abilities of HELM for fault detection and isolation. The two simulated case studies compared different methods including (1) a one-class classifier ELM alone, (2) a Principal Component Analysis (PCA) with a one-class classifier ELM, (3) a Support Vector Machines (SVM), (4) a PCA with a SVM and last (5) a Deep Belief Network.

On the same two case studies, we compare in the following the results of HELM and FHSNN. To do so, we fix the number of neurons in the Auto-encoder layer, as it is where the difference between HELM and FSHNN lies, and do a grid search on other hyperparameters for both HELM and FSHNN independently. In order to fix the number of neurons in the FSHNN, we initialise a population of features, five times bigger than the number of desired neurons, and search the right λ_H in Algorithm (2) that would select exactly this desired number of neurons. This requires 5 iterations on average.

Hyper-parameters for both models are:

- λ_{AE}: the regularization coefficient in the autoencoder

- λ_{1C}: the regularization coefficient in the one-class classifier
- nAE: the number of neurons for the autoencoder
- n1C: the number of neurons for the last layer one class classifier
- γ: the factor used in the decision rule (11)

In both case studies, the methodology for generating the datasets that resemble real condition monitoring applications consists in generating 5 "base" signals from which 200 sensors will be simulated. The 5 base signals represent the inner dimension of the dataset. Then, a random Gaussian noise of 1% is added to the readings. Finally, faults will impact either one base signal or the reading directly. This approach imitates "real" fault types affecting an entire subsystem of fault types only affecting the measurements.

For both simulated case studies, the signals before the fault are split into training, validation and testing data. The training contains 7000 samples, validation and non-faulty testing 1000 each. Faulty sets contains 1000 samples per fault generated.

The validation dataset is used to estimate the detection threshold as per Equation (11). Among the testing data, the non-faulty set enables the computation of the reconstruction error (with the root mean square error (RMSE) and of the *True Negatives* (TN) (complementary to the *False Positives* (FP)), while the *True Positives* (TP), complementary to the *False Negatives* (FN), are obtained from the datasets with faults.

For each dataset and each fault, if the threshold in Equation (11) is exceeded at least once, then it is considered as TP (if the dataset has a fault) or as FP otherwise.

In all case studies, the results are discussed and quantified with the following indicators:

- Accuracy per fault:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{2 \cdot N_{\text{exp}}} \cdot 100 \tag{12}$$

  where N_{exp} is the number of times the experiment has been repeated (100 times in each case).

- Isolation: Once the fault has been detected, we propose to compute for each signal its residual before and after the first sample detected as faulty. The signals with the biggest residual differences are those considered as isolated. Results show the percentage of signals correctly isolated.
- Reconstruction: From the non-faulty testing data, it is possible to measure the reconstruction error on the reconstructed signals using the auto-encoder. This is done using the root mean square error (RMSE).

Table 1. Accuracy per fault comparison for the random signal readings case study for (H) HELM and (F) FSHNN. 6 faults are tested and the best accuracies are summarised for different number of neurons used in the auto-encoder (nAE).

Fault n^o	1		2		3		4		5		6	
nAE	(H)	(F)	(H)	(F)	(H)	(F)	(H)	(F)	(H)	(F)	(H)	(F)
3	77	56	92	57	75	57	61	57	67	60	60	57
5	78	60	97	63	77	57	58	61	70	67	58	61
7	81	65	98	70	80	63	60	63	69	75	63	69
10	78	65	98	76	83	66	64	66	73	83	60	72
15	82	68	98	86	80	69	62	68	72	89	60	79
20	81	71	98	91	83	70	65	71	76	89	61	80
30	80	75	96	98	81	73	66	73	90	97	65	86
40	80	76	98	98	80	76	71	74	93	98	73	86
50	83	77	99	95	86	78	75	79	93	98	75	89
70	83	73	100	93	84	75	80	82	97	98	78	91

4.1. Random Signal Readings Case Study

In (Michau, Yang et al., 2017) the case study consisted in the generation of 5 "base" signals. These base signals are drawn according to a uniform distribution in $[0, 1]$. Then, 200 sensors are simulated: each sensor performs a random reading of one of these base signals according to an internal reading function $f_s(X^{\text{Base}}) = \alpha_s \cdot X_i^{\text{Base}} + C_s + \epsilon$ where s is the sensor ID, α_s is the reading amplitude, C_s is a sensor dependant constant and ϵ is an additive random Gaussian-noise of 1%.

At a given time step, a fault is simulated which can impact either one base signal in X^{Base} (Fault (1) to (3)) or the sensors themselves (Fault (4) to (6)). Simulated faults are:

- (1) and (5): *20% amplitude change*. The signal amplitude is multiplied by 1.2
- (2) and (6): *50% amplitude change*. The signal amplitude is multiplied by 1.5
- (3) and (7): *20% stepwise deviation*. A constant value computed as 20% of the signal amplitude is added

The experiments are run 100 times.

Results for this case study are summarised in Figure 3 and Table 1. In Table 1, the best accuracies achieved by searching the best set of hyper-parameters independently for each fault for both models ((1) HELM, (2) FSHNN) are gathered. It is interesting to note that HELM is performing overall better on faults (1) to (3), impacting the base signals, while FSHNN is performing better on faults (4) to (6) impacting the signals directly. This is also illustrated in Figure 3a and 3b. On these figures, the set of hyper-parameters maximising the average accuracy of the three faults ((1) to (3) and (4) to (6)respectively) is looked for and the resulted are plotted. In Figures 3c and 3d, the isolation performances are illustrated demonstrating that FSHNN features are more useful as they enable the isolation of a higher number of impacted signals (100% of isolation for the faults impacting signals directly). This is supported by Figure 3e, showing that the reconstruction is much better using FSHNN auto-encoder.

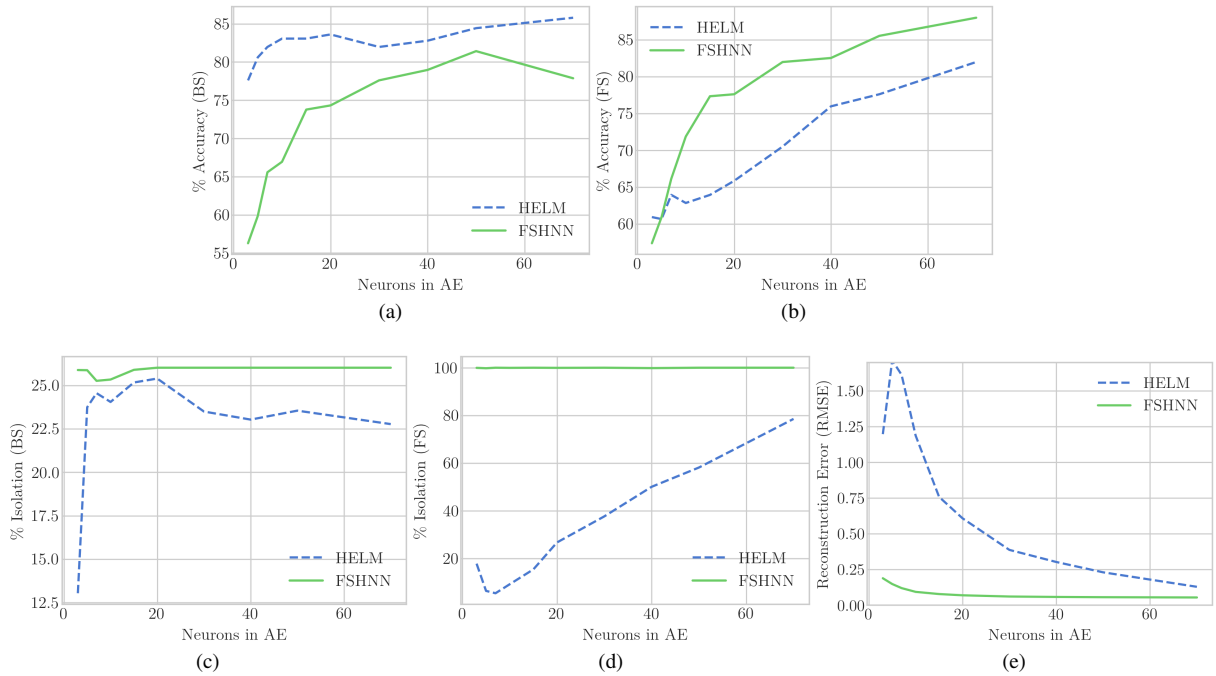

Figure 3. Random Signal Readings case study: Comparison of HELM and FSHNN. (a) average accuracy for faults (1) to (3) impacting the base signals only, (b) average accuracy for faults (4) to (6) impacting the readings, (c) average isolation for faults (1) to (3), (d) (c) average isolation for faults (4) to (6), (e) reconstruction RMSE of the healthy signals.

4.2. Linear Combination of Random Piecewise Linear Signals

The second case study is similar to that presented in (Michau, Palmé & Fink, 2017). It also consists in the generation of 5 base signals. This time, to simulate signals with high dimensionality and high correlation, the base signals are randomly constructed piecewise linear. The random piecewise linearity is used to prevent the possibility for HELM to simply learn temporal patterns. Using random linear combinations of signals pairs (random pairs with random weights), 200 readings are created. At fault, one of the two signals of the pair changes for one of the other base signals. This simulates a fault impacting one part of the system only and changing partially the correlations within the readings.

Results are summarised in Figure 4 and Table 2. As seen in Figure 4, the accuracies achieved both by HELM and FSHNN are in competition, and do not show any statistical differences. This generalises to both the reconstruction and the isolation performances when more than 30 neurons are in use in the auto-encoder. Yet, when the number of neurons is closer to the real inherent dimensionality of the problem (5 base signals), it is where FSHNN has an advantage. Its features are of higher value both for reconstruction and isolation. FSHNN reaches performances close to the best achieved with 10 neurons only.

In addition to the performance of the algorithms, Table 2 also shows the number of neurons used in the one-class classifier layer for the model that achieved best results. It is interesting to note that HELM seems to require as many neurons as possible (500 was the maximum number of neurons tested) while FSHNN achieve its best performances with a lower number of neurons. This observation also supports the expectation that the features used as inputs of the one-class classifier are more informative for the classifier when stemming from FSHNN.

Table 2. Results comparison for the LCRPLS case study for (H) HELM and (F) FSHNN. nAE is the number of neurons in the auto-encoder and n1C the number of neurons in the one-class classifier for the model with best accuracy.

nAE	n1C		Accuracy		Isolation		Reconstruction	
	(H)	(F)	(H)	(F)	(H)	(F)	(H)	(F)
3	500	50	54	62	56	63	0.55	0.15
5	500	100	53	74	55	72	0.78	0.11
7	500	100	83	77	55	84	0.75	0.09
10	500	200	88	87	70	92	0.50	0.07
15	500	300	89	86	87	94	0.29	0.06
20	300	300	92	91	92	95	0.21	0.06
30	500	200	92	89	98	97	0.12	0.05
40	500	300	92	91	100	96	0.08	0.05
50	500	300	93	94	100	95	0.07	0.05
70	500	300	93	94	94	94	0.07	0.05

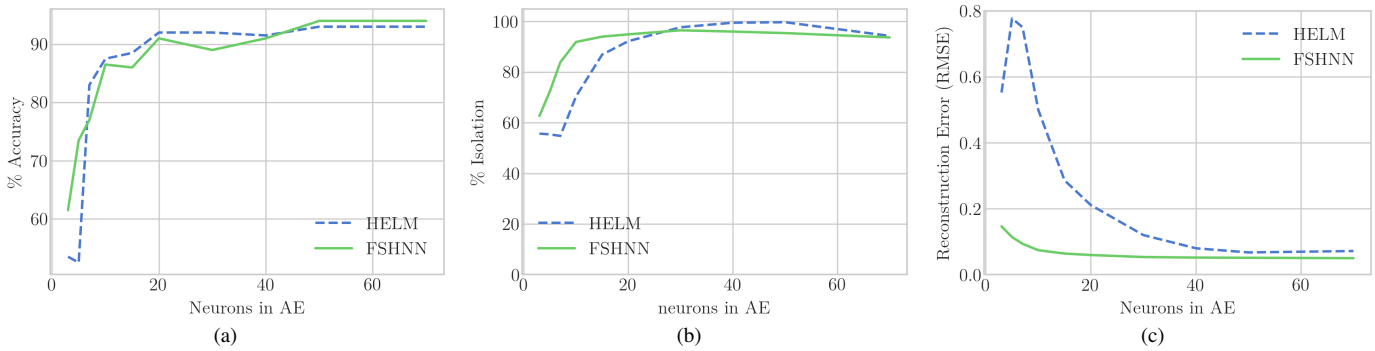

Figure 4. Linear Combination of Random Piecewise Linear Signals (LCRPLS) : (a) Accuracy (b) Isolation of faulty signals (c) Reconstruction RMSE over all signals in healthy conditions.

4.3. Real application of fault detection in a generator

Both in (Michau, Palmé & Fink, 2017; Michau, Yang et al., 2017), the results where tested on a real case study, a H_2 cooled generator from a electricity producing plant. The generator is working on nominal load, with changing operating conditions and thus, variations in the parameters. The evaluated dataset consist of nine months (275 days) and includes data from 320 sensors, measuring physical quantities. They can be grouped in 5 families: Partial discharge monitoring, Rotor shaft voltage, Rotor flux, Stator end winding vibration, Stator bar water temperature.

The observation period corresponds to 275 days of operation. In particular, after day 247 (approximately eight months of operation), an upper-level fault occurred. Experts having analyzed the data concluded *a posteriori* that some signals started to show abnormal behavior at day 169, consequence of a lower level fault. The generator is operated in base load, meaning that the plant is operated at full power. The dataset consists of $K = 55\,774$ observations and $D = 320$ dimensions.

As the fault beginning was confirmed at day 169, we decided to use the data up to day 120 for both training and validation, with a random sampling $94\%/6\%$. The rest of the measurements, until the fault and after, are used for false and true positive quantification respectively. The dataset is hence split as follows: $K^{\text{Train}} = 22\,017$, $K^{\text{Val}} = 1\,406$ and $K^{\text{Test}} = 4\,524 + 27\,827$.

By running 100 experiments for each set of hyper-parameters, we could find with both models, independently of the number of neurons in the auto-encoder 100% detection accuracy. The reconstruction error and the output of both models are compared in Figure 5. As for the simulated case studies, FSHNN demonstrates similar a performance for fault detection but a much better performance on signals reconstruction.

For the isolation, the ground truth is not known and it is therefore difficult to evaluate the performance of both models. In order to give an idea of isolation performance, the isolated signals found by both models are collected over 100 experiments. For each model and for increasing number of neurons in the auto-encoder layer, the best set of hyper-parameters with respect to the accuracy is selected. The 10 first isolated signals are collected. Overall, 63 out of 310 signals are identified by at least one model, and 8 of those are isolated in more than 80% of the cases. We assume therefore that those 8 signals are indeed where the fault is originating. Then for each model, and each experiment, we measure the ratio of these 8 signals that have indeed been isolated by each model. The results are represented in Figure 5b. In accordance with the simulated case studies, FSHNN demonstrates a higher performance, in particular with a low number of neurons in the auto-encoder. Interestingly, its performance is almost not depending on the number of neurons in the auto-encoder, in particular when this number is very small. This supports the expectation that the features are informative and that their selection brings valuable information to the model that enables to distinguish between signals that were affected by the fault and those that did not experience any deviation from the "normal" system behaviour.

4.4. Results Discussion

On the basis of the present analysis, the benefits and disadvantages of FSHNN can be summarised as follows:

Detection: FSHNN and HELM achieve often similar results (e.g., Fig. 4a, 5). Over the many experiments performed, FSHNN is often performing slightly better than HELM when the number of neurons is very low (e.g., Fig 3b or Fig 4a with 3 neurons), but this depends on the case study (e.g., Fig 3a is a counter-example). This comparison of detection performances has been led by optimising, in each case and for each model the best set of hyper-paramaters with respect to the performances. And if by doing so, HELM and FSHNN reach similar detection performances, FSHNN often does so with a lower number of neurons needed in the one-class classifier.

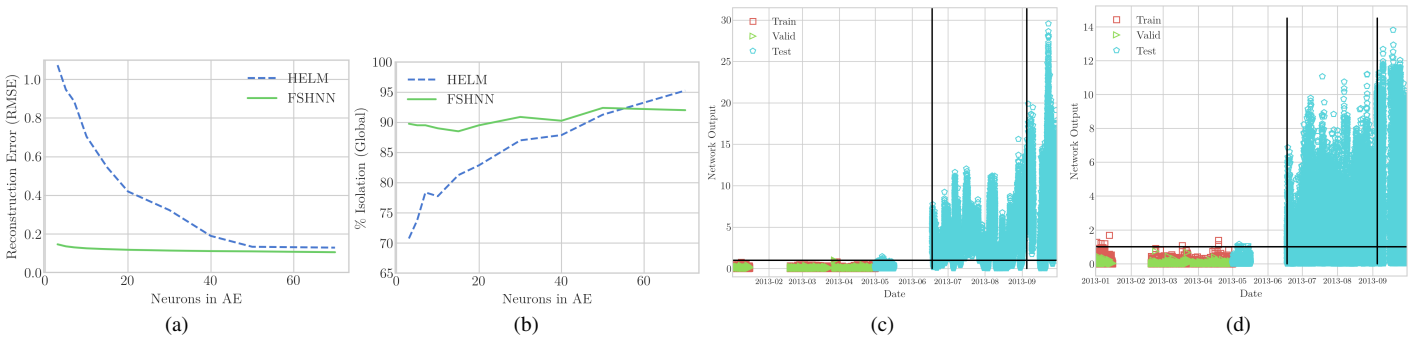

Figure 5. Generator Case Study : (a) Reconstruction performances, (b) Isolation, (c) HELM output, (d) FHSNN output.

Reconstruction and Isolation: Due to the higher reconstruction performances (e.g., Fig. 3e, 4c, 5a), FSHNN is often outperforming HELM for signal isolation (e.g., Fig.3d). This is also the case where HELM had better detection performances (e.g., Fig.3c), and in particular when the number of neuron is low (e.g., Fig. 4b).

Computation cost: There is not any noticable computational difference when training a HELM or a FSHNN. Both were trained using FISTA for the auto-encoder (but with a different regularization) and a ridge regression for the one-class classifier. One difference lies in the iterations needed to find the right regularisation parameter for an *a-priori* fixed number of neurons. Yet, this was done for comparison purposes only and might not be relevant in real applications. In addition, to account for randomness in HELM results, in particular with a low number of neurons in the auto-encoder, it has been proposed in (Michau, Yang et al., 2017) to average several runs of HELM (e.g., 5). This is computationally-wise in line with the 5 iterations of FSHNN needed on average to find the right regularisation parameter. For a comparison of HELM (and therefore FSHNN) against other common approaches, the reader is refereed to (Michau, Palmé & Fink, 2017; Michau, Yang et al., 2017).

Impact of the number of neurons in the auto-encoder layer: In many cases, it looks like HELM is catching up with FSHNN performances when the number of neurons in the auto-encoder layer increases. This is actually to be expected for two reasons. First, HELM uses random features. When the number of features becomes big compared to the inherent dimensionality of the problem, the likelihood to have informative features increases greatly, making the careful selection of features unnecessary. A one-class classifier with many neurons will be able to extract the relevant information.

Second, the group-lasso regularisation needs to be very strong in order to select a small fraction of the features. This limits the potential of FHSNN as the ℓ_{21} norm both fosters sparsity but also small weights. In short, the stronger we select, the less accurate the encoding will be. If this equilibrium is to the advantage of FHSNN when a low number of features is imposed, as it is crucial to have informative features, this advantage disappears with more features, when the likelihood of having good features anyhow increases greatly.

Yet, if it is tempting to always use HELM with a high number of features, one needs to consider that, if FSHNN can achieve similar performances with fewer features, that means that many features in the HELM are probably irrelevant and contain only little information. If the aim, in practice, is to have the features analysed by an expert, this gives HELM a disadvantage.

5. CONCLUSION

In this paper, we proposed a new Feature Selecting Hierarchical Neural Network (FSHNN). The proposed approach combines the benefits of a fast and efficient training process and a targeted feature selection resulting in better representation of the condition monitoring signals in a lower dimensional feature space. The lower dimensional representation of the input signals is not achieved by contracting the input signals to a lower dimensional feature representation directly, instead, a strong Group-LASSO regularization is applied on a large number of randomly generated features to select as few features as possible. Thereby it achieves a low dimensional representation of the input signals with a small number of most informative features. The proposed approach is similar to the feature generation and feature selection approach commonly applied in prognostics and health management applications, with the main difference that features are generated and selected automatically: the information contained in the signals is learned without the need of manual feature engineering. However, contrary to the previously proposed feature learning approaches based on HELM, FSHNN reaches a more informative latent space due to performing the feature selection process typically performed by the experts. Even though the extracted features have not been evaluated further in this paper, the higher degree of information contained in the learned

features is supported by the improved performance on fault isolation.

The case studies presented in this paper, confirm the very good fault detection abilities of the proposed algorithm, which is at par with the previously proposed HELM algorithm that was already achieving an excellent fault detection and isolation performance on previously performed case studies. It outperforms, in addition, HELM for signal reconstruction and isolation. The evaluation of FSHNN demonstrates the excellent ability of the algorithm to select the most meaningful features in the input space: the features bring better reconstruction, isolation and necessitate less neurons in the one-class classifier for achieving detection performances in line with HELM.

The feature selection ability of the proposed FSHNN algorithm could be particularly useful within ensemble approaches where several sets of specialised features for the different operating conditions could be used to achieve a robust fault detection and isolation performance also under varying operating conditions without the need to develop dedicated algorithms for each of the operating conditions separately.

An additional further research direction could be the monitoring of the change of the features that are selected over time (after retraining) and monitoring the change in the importance of the features. This could provide insights into the process evolution over time.

Last, the question of feature interpretability or quality with criteria on the features themselves, and not *a posteriori* based on the detection, reconstruction, or isolation performances, remains also a relevant question for future research.

REFERENCES

Beck, A. & Teboulle, M. (2009, January 1). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, *2*(1), 183–202.

Bertsimas, D., King, A. & Mazumder, R. (2016). Best subset selection via a modern optimization lens. *The annals of statistics*, *44*(2), 813–852.

Briceño-Arias, L. M., Combettes, P. L., Pesquet, J.-C. & Pustelnik, N. (2011, September 1). Proximal Algorithms for Multicomponent Image Recovery Problems. *Journal of Mathematical Imaging and Vision*, *41*(1-2), 3–22.

Chambolle, A., Dossal, C. et al. (2014). How to make sure the iterates of FISTA converge.

Friedman, J., Hastie, T. & Tibshirani, R. (2001). *The elements of statistical learning*. Springer series in statistics New York.

Gugulothu, N., Tv, V., Malhotra, P., Vig, L., Agarwal, P. & Shroo, G. (2017). Predicting Remaining Useful Life using Time Series Embeddings based on Recurrent Neural Networks. *10*.

Hastie, T., Tibshirani, R. & Tibshirani, R. J. (2017, July 26). Extended Comparisons of Best Subset Selection, Forward Stepwise Selection, and the Lasso. arXiv: 1707.08692 [stat]

Hocking, R. R. & Leslie, R. N. (1967). Selection of the Best Subset in Regression Analysis. *Technometrics*, *9*(4), 531–540. JSTOR: 1266192

Huang, G., Huang, G.-B., Song, S. & You, K. (2015, January). Trends in extreme learning machines: A review. *Neural Networks*, *61*, 32–48.

Huang, G.-B., Chen, L., Siew, C. K. et al. (2006). Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Networks*, *17*(4), 879–892.

Huang, G.-B., Wu, J. & Wunsch, D. C. (2018, February 14). Hierarchical extreme learning machines. *Neurocomputing*. Hierarchical Extreme Learning Machines, *277*, 1–3.

Ince, T., Kiranyaz, S., Eren, L., Askar, M. & Gabbouj, M. (2016). Real-Time Motor Fault Detection by 1-D Convolutional Neural Networks. *IEEE Transactions on Industrial Electronics*, *63*(11), 7067–7075.

Khan, S. S. & Madden, M. G. (2009, August 19). A Survey of Recent Trends in One Class Classification. In *Artificial Intelligence and Cognitive Science* (pp. 188–197). Lecture Notes in Computer Science. Irish Conference on Artificial Intelligence and Cognitive Science.

Man, Z. & Huang, G.-B. (2016, February 1). Guest editorial: Special issue on Extreme learning machine and applications (II). *Neural Computing and Applications*, *27*(2), 253–254.

Michau, G., Palmé, T. & Fink, O. (2017, October). Deep Feature Learning Network for Fault Detection and Isolation. In *Annual Conference of the Prognostics and Health Management Society 2017*, St. Petersburg, Florida.

Michau, G., Palmé, T. & Fink, O. (2018, July 2–5). Fleet PHM for Critical Systems: Bi-level Deep Learning Approach for Fault Detection. In *Proceedings of the fourth european conference of the prognostics and health management society 2018*, Utrecht, The Netherlands.

Michau, G., Yang, H., Palmé, T. & Fink, O. (2017, April). Feature learning for fault detection in high-dimensional condition-monitoring signals. *submitted*, 1–10.

Pecht, M. & Jie Gu, J. (2009). Physics-of-failure-based prognostics for electronic products. *Transactions of the Institute of Measurement and Control*, *31*(3-4), 309–322.

Pustelnik, N., Chaux, C. & Pesquet, J.-C. (2009, November 8). Parallel proximal algorithm for image restoration using hybrid regularization – extended version. *IEEE Transactions on Image Processing*, *20*(9), 2450–2462.

Sateesh Babu, G., Zhao, P. & Li, X.-L. (2016). Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life. (pp. 214–228).

Tang, J., Deng, C. & Huang, G.-B. (2016). Extreme learning machine for multilayer perceptron. *IEEE transactions on neural networks and learning systems*, *27*(4), 809–821.

Tinga, T. & Loendersloot, R. (2014). Aligning PHM, SHM and CBM by understanding the physical system failure behaviour. PHM society.

Vachtsevanos, G., Lewis, F., Roemer, M., Hess, A. & Wu, B. (2006). *Intelligent fault diagnosis and prognosis for engineering systems*. John Wiley & Sons, Inc.

Yang, H., Fink, O. & Palmé, T. (2016). Online sequential extreme learning machines for fault detection. In *Prognostics and Health Management (ICPHM), 2016 IEEE International Conference on* (pp. 1–8). IEEE.